Journal of Computer and Knowledge Engineering, Vol. 6, No. 1, 2023. (71-79)
DOI: 10.22067/cke.2023.79099.1070

Trace2Vec-CDD: A Framework for Concept Drift Detection in Business Process Logs using Trace Embedding*

Research Article

Fatemeh Khojasteh[1] Behshid Behkamal[2] Mohsen Kahani[3] MahsaKhorasani[4]

Abstract: Business processes are subject to changes during their execution over time due to new legislation, seasonal effects, etc. Detection of process changes is alternatively called business process drift detection. Currently, existing methods unfavorably subject the accuracy of drift detection to the effects of window size. Furthermore, most methods have to struggle with the problem of selecting appropriate features specifying the relations between traces or events. This paper draws on the notion of trace embedding to propose a new framework for automatic detection of suddenly occurring process drifts. The main contributions of the proposed approach are: i) It is independent of windows; ii) Trace embedding that is used for drift detection makes it possible to automatically extract all features from relations among traces; iii) As attested by synthetic event logs, this approach is superior to current methods in terms of accuracy and drift detection delay.

Keywords: Concept Drift, Process Changes, Process Mining, Word Embedding

1. Introduction

Modern business processes are handled by information systems. Information systems produce event logs, which are sources of information about the actual processes. It is typical for a business process to change over time, which may be due to factors such as substantial changes in supply and demand, seasonal reasons, etc. These changes have considerable impacts on the process costs and efficiency.

Detection of business process drifts can be considered as a variant of the general issue of concept drift detection, which has received much attention in data mining and machine learning. Experts in such areas use the term "concept drift" when the distribution of a variable has experienced a change [1]. In process mining, however, the challenge is detecting more complex changes, such as changes in the process models that describe choices, loops, cancellations, and concurrency.

Therefore, drift detection methods used in data mining cannot directly be applied to detect drifts in business processes. Based on the definition of concept drift in process mining, wherever the traces before and after a specific point differ in characteristics, a business process drift has occurred [2].

In the area of process mining, various techniques have been proposed to detect concept drift, most of which perform statistical comparisons between pairs of windows. Some methods are based on fixed windows [3, 4, 5, 6, 7]. During the process, these windows keep their size unchanged. Other methods use adaptive windows in which the size of windows changes during the process [2, 8, 9, 10, 11]. Adaptive windows provide higher accuracy than fixed windows. However, the initial size of an adaptive window has to be determined log by log.

On the other hand, most of the current methods rely on selecting features that characterize traces. Features such as relation type count (RC), relation entropy (RE), window count (WC), and J measure [2, 3] are typical examples. The point is that good levels of accuracy sometimes require the user to be knowledgeable about the features of drifts. If the user is not knowledgeable enough, unsuitable features may be selected, which leads to a failure in identifying some kinds of features.

The goal of this study is to solve the aforementioned problems using the idea of *trace embedding*. This concept has already been introduced in [12], which is based on the notion of *paragraph2vec*. Here, we presented a new definition of trace2vec based on *word2vec* [13]. We extract the features from the relations between traces and identify sudden process changes based on similarities between the vectors. Thus, the main novelty of this work is the elimination of windows, which results in more accurate detection of process drifts. The main contributions of this study are:

1. The idea of trace embedding is applied for concept drift detection. This is exploited in the automatic extraction of features from traces, the straightforward comparison of vectors of traces and change detection;
2. The Fourier transform is used to omit noise and outlier traces in the log;
3. Unlike the existing methods, the proposed approach is window-independent;
4. The artificial logs of [8] are used to determine the accuracy of our approach.

They show our approach to be remarkably more accurate than the state-of-the-art methods in terms of F-score and drift detection delay.

The remainder of this paper is organized as follows.

---

* Manuscript received: 2022 October 8, Revised, 2023 April 15, Accepted, 2023May 24.

[1], Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.
[2] Corresponding author. Assistant Professor, Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran, **Email**: behkamal@um.ac.ir.
[3] Professor, Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.
[4] Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.

Section 2 includes the literature review. In section 3, our proposed approach, Trace2Vec-CDD, is discussed in detail. Then, the proposed framework is empirically evaluated in Section 4. Finally, Section 5 deals with the conclusion and future directions.

2. Related works

In this section, the existing methods for concept drift detection from various aspects are reviewed. To this end, these aspects are introduced and the related works are then classified according to these dimensions.

2.1. Detection method

State-of-the-art methods can detect drifts in either an offline or online setting. In the offline setting, the whole log must exist, whereas in the online one, concept drifts will be detected by sequential monitoring of the logs of a system and react to the changes in an online, almost real-time way [6, 9, 14, 15]. The method [6] proposes one of the methods that use online setting. [9] is another online approach that proposes an event-based method that performs well with processes in which there is a high ratio of distinct executions to the total number of executions in the log. All other existing methods work in an offline setting. [15] presented an online technique for detecting drifts. For that, trace distances are calculated by comparing them to a global model that represents the current state of the process. Hence, a density-based clustering algorithm is applied to distribute the instances in the feature space. Finally, the discovery of new clusters represents the detection of new concepts in the stream, i.e., concept drift.

2.2. Type of window

Most of the existing methods for business process drift detection use static or dynamic windows. The first group of methods use static windows [7,4, 6, 3, 5]. In such studies, the accuracy of the drift detection method is dependent on the size of the window. The second group of methods use adaptive window [2, 8, 9, 16, 10, 11, 14, 17]. The idea of using adaptive windows is to set minimum and maximum values for the size of the window and increase the minimum value untila change is detected or the window size reaches the maximum size limit. Thus,if the minimum size is too small, noises may be detected as drifts. On the other hand, if the maximum size is too large, some drifts may not be detected.

2.3. Perspective

There are three approaches to analyzing process models [3]: 1. Control flow, which is concerned with behavioral and structural changes in a process model; 2. Data, where changes refer to the changes in the production and consumption of data and the impact of data on the routing of cases; and 3. Resource, which is the changes in resources, their roles, and organizational structure. Most of the previous methods have considered the control-flow perspective of process models. The only solution that considers both the control-flow and data perspectives is the one suggested by [5]. In order to identify change points, the similarity between two consecutive windows is compared using the Markov clustering algorithm.

2.4. Type of drift

Based on the classification presented in [18], there are four types of drifts: 1. Sudden drift, in which a new process replaces an existing one; 2. Gradual drift, in which parts of both new and old processes coexist for a period of time; 3. Recurring drift, when a set of processes re-appear after some time; and 4. Incremental drift, in which a new process is substituted for an existing process via plenty of minor incremental changes.

Most of the previous methods can detect sudden drift, while few of them can detect other types of drift too. For example, [3, 19] addressed the detection of sudden drifts and certain types of gradual drifts in process mining. The method proposed by [2] considers windows at different time scales to detect recurring drifts. This method does not work for logs involving many process variants. The authors of [4] claimed that their method can detect all sudden, gradual, and recurring drifts using fixed windows. The approach clusters traces based on the distance between pairs of activities. [8] proposed an automatic approach in order to build a "*run*" from eachtrace in order to detect sudden drifts in two sequential adaptive windows.

In [11] the researchers extended their previous method [8] to detect not only sudden drifts but also gradual drifts. They believe that gradual drifts will appear in the form of two consecutive sudden drifts. They applied a statistical test to determine whether the detected sudden drifts are separate changes or a single gradual drift. A limitation of the method is the requirement to re-size the adaptive window to arrive at a trade-off between accuracy and drift detection delay. Moreover, [17] presented two new algorithms to detect incremental, sudden, recurring, and gradual drifts. The first algorithm creates the process history and discovers new viable models based on conformance and a sliding window approach. The second algorithm determines concept drifts based on the synthesized process histories.

Table 1 provides a feature-based comparison of previous studies based on the aforementioned aspects. As shown in this table, most of the methods use the window, fixed or adaptive. Moreover, the concept of embedding has not been applied to drift detection so far. In this work, we propose an embedding-based approach for concept drift detection that is window independent.

Table 1. Feature-based comparison of the related works

Ref.	Detection method		Type of windows		Perspective			Type of drifts			
	Offline	**Online**	**Fixed**	**Adaptive**	**control-flow**	**Resource**	**Data**	**Sudden**	**Gragual**	**Recurring**	**Incremental**
[7]	*		*		*			*			
[4]	*		*		*			*	*	*	
[6]		*	*		*			*			
[3]	*		*		*			*	*		
[5]	*		*		*		*	*			
[2]	*			*	*			*	*	*	*
[8]	*			*	*			*			
[9]		*		*	*			*			
[20]	*		*		*			*			
[16]	*			*	*			*	*		
[10]	*			*	*			*	*		
[11]	*			*	*			*	*		
[17]	*			*	*			*	*	*	*
[14]		*		*	*			*			
[21]	*		*		*		*		*	*	
[15]		*	*		*			*	*		
Proposed method	*		window -independent		*			*			

3. Proposed approach

In order to detect concept drifts in business process logs and overcome the limitations of the previous approaches, we use trace embedding in the detection of process drifts. The general architecture of our proposed approach is illustrated in Figure 1, which consists of four main phases. Below, the phases are separately discussed in detail.

3.1. Modeling trace to vector

In this step, we use trace embedding to automatically extract features from the relations between traces.

By interpreting process event logs as texts and process traces as words, we can apply the idea of trace embedding to this step. Algorithm 1, which is used in this step, receives a list of traces LT as its input and outputs a set of traces vectors TV.

At the beginning of the algorithm, we need to consider traces as words and change each trace to a single word by omitting any spaces, underlines, and so on. LWT = ConvertTrace2Word(t) t LT , where $ConvertTrace2Word$ is a function that changes a trace t to a single word, and LT is a list of traces, which is ordered on the basis of the timestamp of the first event (Line 2 in Algorithm 1).

The TraceEmbedding function is trained by the list of traces, which results in the representation of each trace in the log as a vector of numerical values in a latent feature space (Line 6 in Algorithm 1). In other words, we transform LWT to a set of trace vectors $TV = Trace\ Embedding\ (t')\ t'\ \ LWT$, where *Trace Embedding* is a function that computes the vector of trace t' in the set LWT using CBOW model.

As mentioned earlier, the idea of *trace2vec* as presented in [12] is based on the notion of *paragraph2vec* introduced by [22], which involves numbers denoting the order of paragraphs. Applying this method in our study causes identical traces that occur in differing positions in the log to have different vectors. This is not desirable in our study. As a result, we presented a new notion of trace embedding.

3.2. Detecting non-co-occurring traces

The objective of this phase is to detect non-co-occurring traces in two steps: Calculating trace similarity and clustering co-occurring traces.

Calculating trace similarity: As mentioned above, the traces are represented through their vector models from the embedding space. Therefore, it is possible to directly calculate the similarity between all pair-wise combinations of trace vectors.

To calculate trace similarity, we use the set of trace vectors *TV* and the set of traces in the event log to produce the similarity matrix *SM* and a list of traces with a minimum co-occurrence. Then, by using the values of similarity, we are able to separate the pairs of traces with the minimum co-occurrence or the maximum distance in the vector space. Such pairs may lead to the occurrence of drifts. The principal assumption is that if the substitution of a process B for a process A has produced a case of drift, there has been a large distance (or a small number of co-occurrence relations) between the set of traces of process A and those of process B. In other words, the trace vectors before the change point differ from the ones after the change point.

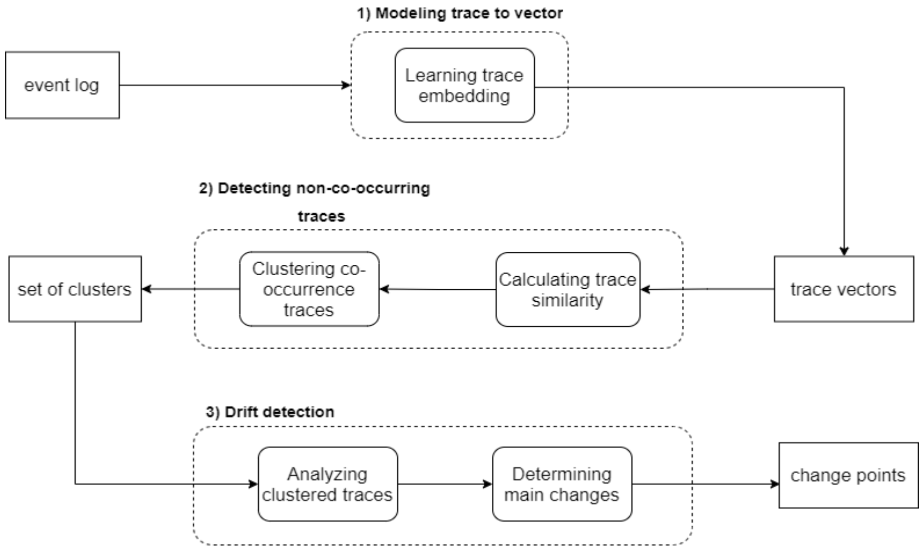


Figure 1. Architecture of the proposed approach

Clustering co-occurring traces: After identifying the set of traces with the minimum co-occurrence, we will use maximum co-occurrence to cluster them. The clustering leads to a situation in which all members of the same cluster are in a similar feature space.

This step receives the similarity matrix created in the previous step as its input and produces a set of clusters as the output. We apply hierarchical clustering with single linkage in which traces cannot be placed in multiple clusters. The threshold of clustering is empirically set to 0.99.

Algorithm 1. Modeling trace to vector

Input : LT: a list of traces ordered on the basis of the timestamp of the firstevent
Output: TV : a set of trace vectors
1 **for** *each* t_i $\in$ LT **do**
2 t'_i = ConvertTrace2Word(t_i);
3 ADD t'_i to LWT;
4 **end**
5 **for** *each* $t'_i \in LWT$ **do**
6 $TV.add$ TraceEmbedding(t_i) ;
7 **end**
8 **return** TV

Algorithm 2. Calculating trace similarity

Input : $TV = v_1, v_2, ..., v_n\}$: a set of trace vectors, $T = t_1, t_2, ..., t_n\}$: aset of traces in L
Output: SM: a similarity matrix, NCT : a list of traces with the minimumco-occurrence
SET NCT to null;
for *each* $v_i \in TV$ **do**
for *each* $v_j \in TV$ **do**
d_{ij}=Similarity(v_i,v_j);
if*CheckNoCoOccurrance*(d_{ij}) **then**
ADD Tvi to NCT ;
ADD Tvj to NCT ;
ADD dij to SMij;
end
end
end
return SM

3.2. Drift detection

In this phase, some special techniques are employed to reduce delays in detecting changes as much as possible. The phase consists of two steps: Clustered trace analysis and determining main changes.

Analyzing clustered traces: This step aims to determine how the members of each cluster are distributed in a log, which leads to the creation of a distribution vector for each cluster. The step receives the output of the former step, i.e., the set of clusters *CL*, as its input and outputs a set of distribution vectors.

In this step, for each cluster, where a member of the cluster appears in the log, we represent it by 1 in the distribution vector associated with the cluster; otherwise, we represent the member by 0.

Figure 2 illustrates how the two clusters (C1 and C2) are distributed in a log that has 2500 cases. Dense areas in Figure 2 indicate the positions of the traces of each cluster in the log.

Determining the main changes: The main objective of this step that uses Algorithm 3 is to determine the change points in the distribution vectors. The set of distribution vectors DV produced in the previous step is input into the algorithm of this step and, as the output, we will have a set of indices of the traces in which drifts have occurred.

In order to determine the frequency regions, the Fourier transform is applied to the distribution vectors (Line 3 in Algorithm 3). The Fourier transform breaks up a signal into its frequency components [23]. In this study, to distinguish the main changes, we need to preserve high frequencies and eliminate low ones. To fulfill this, low-pass filtering is applied (Line 4 in Algorithm 1). In other words, the traces that have been incorrectly detected as co-occurring traces will be eliminated by applying low-pass filtering. Then, the distribution vectors are transferred back to the time domain (Line 5 in Algorithm 3). Afterwards, the rate of changes is determined through differentiation (Line 6 in Algorithm 3). The positions associated with the highest rate of changes or notably distinguishable peaks are specified as indices of the traces in which a drift has occurred (Line 7 in Algorithm 3). For instance, in Figure 3, 9 drifts will be detected at the indices of 250, 500, 750, 1000, 1250, 1500, 1750, 2000, and 2250, all of which are positions of notably distinguishable peaks.

In summary, by applying trace embedding and also various techniques such as hierarchical clustering and Fourier transform, we managed to propose an approach marked by the following innovative characteristics: It automatically extracts features from traces and events. Besides, since it uses no windows for detecting drifts, it is not sensitive to window size.

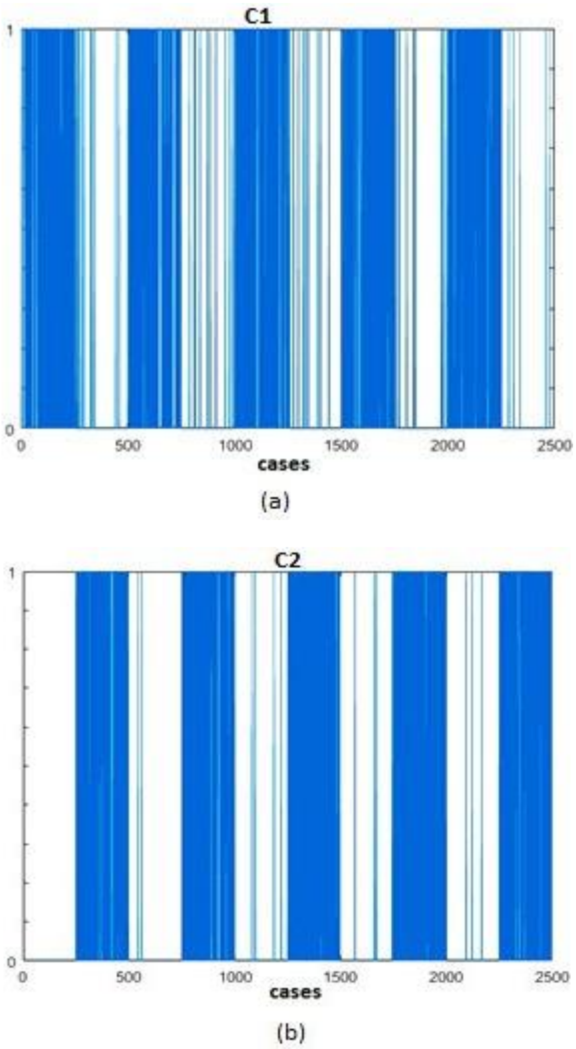


Figure 2. Distribution of each cluster in a log

Algorithm 3. Determining main changes

Input: $DV = dv_1, dv_2, \ldots, dv_n\}$: a set of distribution vectors
Output: D: a set of indexes of the traces in which drifts have occurred
SET *drifts* to *null;*
for *each* $dv_i \in DV$ do
fft = fft *(dvi);*
lpf =LowPassFilter *(fft);*
ifft= ifft *(LPF);*
diff = Differentiate *(ifft);*
d_i =ReportDrift *(diff);*
ADD *di* to *D*;
end
return D;

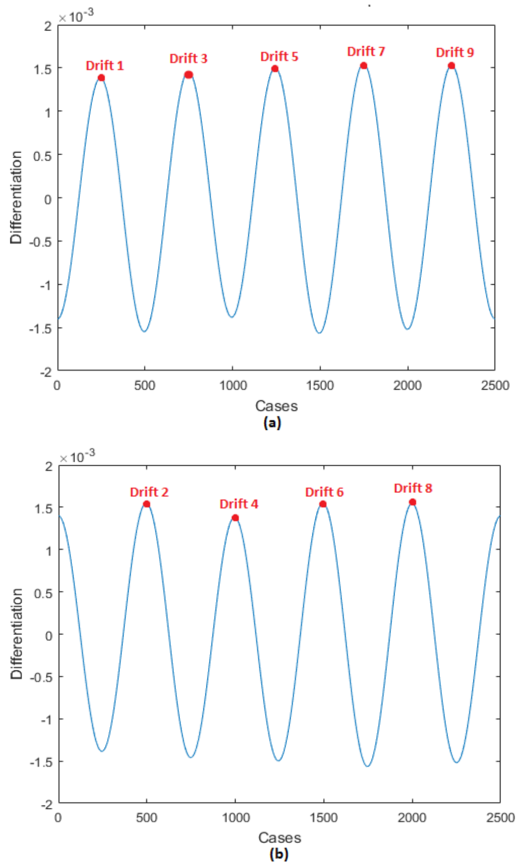


Figure 3. (a) The rate of changes in the distribution vector of cluster C1; (b) The rate of changes in the distribution vector of cluster C2. Remarkable peaks are change points.

4. Evaluation

In this section, the evaluation metrics are discussed, and the evaluation of the proposed approach is presented. Then, the performance of our method is evaluated in comparison with two different categories of state-of-the-art methods.

In order to implement the proposed approach, Deeplearning4j (https://deeplearning4j.org) was used to model traces and activities as vectors. We use the CBOW model with the following parameters: Window size = 20 (based on the mean length of the traces); number of iterations = 10; and vector dimension = 100. The remaining parameters have the default values as proposed in [24].

Hierarchical clustering algorithm was implemented in Java. Moreover, Matlab tool was used for Fourier transform computing and applying low-pass filtering.

4.1. Evaluation metrics

In terms of evaluation metrics, we compare the performance of our work with the state-of-the-art methods using two measures: F-score and mean delay.

In the case of our study, True Positive is the number of drifts that were correctly detected, False Positive specifies the number of drifts that the method incorrectly detected, and False Negative is the number of drifts that the method was not able to detect.

In other words, F-score specifies whether our approach has correctly identified drifts in an event log or not. Besides, we calculate the mean delay, which is the distance between actual drift points and detected drifts.

4.2. Data set

We initially describe the data set used in the experiment and then the results of our method are compared with the methods proposed in [3], [8], and [10].

To evaluate our approach, the synthetic logs published by [8] were used. The base model of these logs, which has 15 activities, involves various control-flow structures. Its BPMN representation, which is about assessing loan applications, is illustrated in Figure 4. The base model was systematically modified to generate drifts. These modifications included 12 simple change patterns organized into three categories: Insertion (I), Resequentialization (R) and Optionalization (O) as shown in Table 2. Moreover, the categories were combined to produce more complex patterns including IOR, IRO, ORI, OIR, RIO, and ROI. Four logs of 2500, 5000, 7500, 10000 traces were produced for each of the 18 simple and complex change patterns. Drifts were injected by switching the drift toggle on and off every 10% of the log. Therefore, any instance of the produced logs included 9 drifts.

4.3. Accuracy evaluation

The accuracy of the proposed approach is evaluated as follows. First, the 18 change patterns discussed in 4.2 were applied to the four log sizes. Then, considering each of the 18 change patterns, the proposed approach was compared against the methods put forward by [8] (called *run*), [10] (called *process-graph*), and [3] (called *Bose*) in terms of the measures of F-score and mean delay. The values resulting for the two measures, averaged over the four log sizes are demonstrated in Figures 5 and 6. Our approach secured the F-score of exactly 1 for all patterns, except for the OIR pattern (0.99), far better than what the methods achieved. Moreover, in terms of mean delay, our method outperforms the *Bose*, *process-graph*, and *run* approaches, except for two change patterns.

On average, our method, *run*, *process-graph*, and *Bose* approaches achieved an F-score of approximately 0.9998, 0.97, 0.94, and 0.701, respectively. Furthermore, in terms of delay, our method managed to achieve a mean delay of about 13 traces, while the *run*, *process-graph*, and *Bose* approaches achieved mean delays of approximately 32, 24, and 47 traces, respectively. Table 3 includes the exact values of F-score and mean delay for each individual change pattern as well as the total average made by each of the four methods.

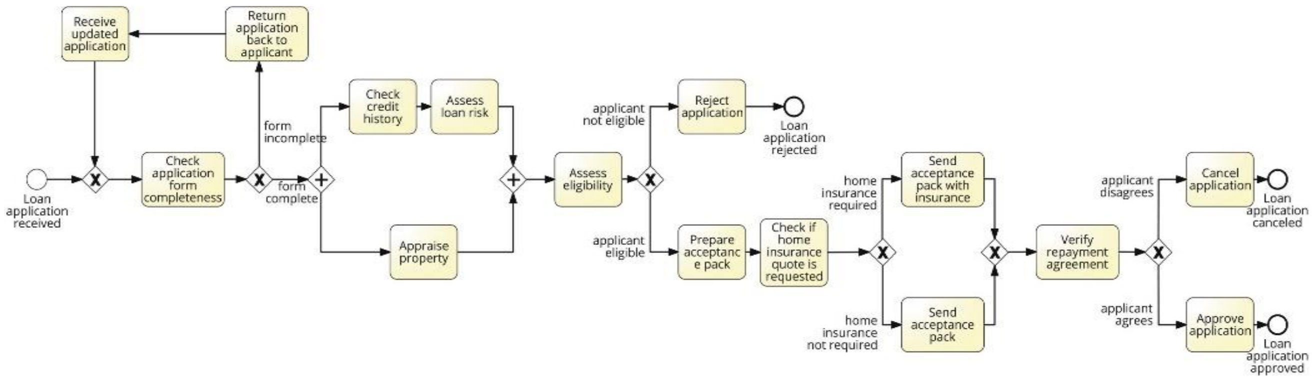


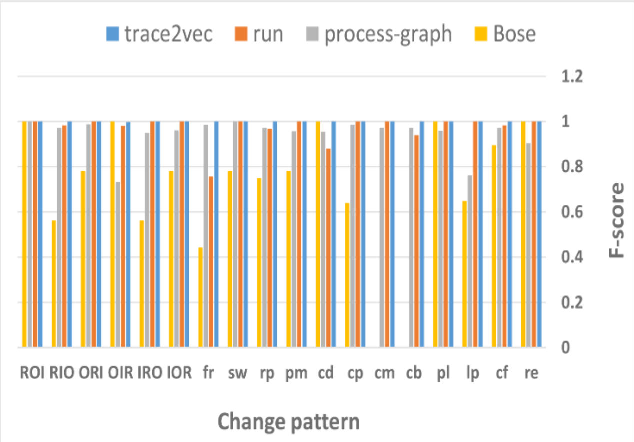
Figure 4. Base BPMN model of loan application process

Table 2. Simple control-flow change patterns

Code	Simple change pattern	Category
re	Add/remove fragment	I
cf	Make two fragments conditional/sequential	R
lp	Make fragment loopable/non-loopable	O
pl	Make tow fragments parallel/sequential	R
cb	Move fragment into/out of conditional branch	O
cm	Move fragment into/out of conditional branch	I
cd	Synchronize two fragments	R
cp	Duplicate fragment	I
pm	Move, fragment into/out of parallel branch	I
rp	Substitute, fragment	I
sw	Swap two fragments	I
fr	Change branching frequency	O

Table 4 shows the results of the statistical tests, i.e., t-tests, on the proposed approaches, *run*, *process-graph*, and *Bose* in terms of F-score and mean delay. In each of the tests, the proposed approach served as the first group. The p-value in all the tests (for both F-score and mean delay) is less than 0.05, except in the case of F-score in the test between *trace2vec* and *run*. This demonstrates that the difference in means is statistically significant at the 0.05 level.

Moreover, the positive upper and lower F-score values and negative higher and lower mean delay values, with the exception of the test between *trace2vec* and *run*, indicate that the suggested methodology produce better average F-score and mean delay values than previous approaches.



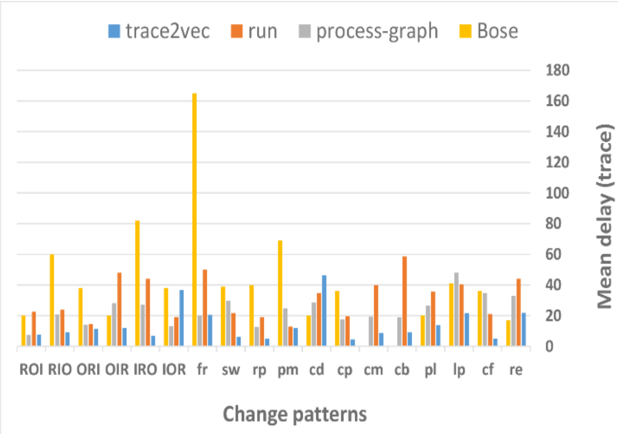


Figure 5. The comparison of F-score values per change pattern Figure 6. The comparison of mean delay values per change pattern

Table 3. The comparison of average F-score and mean delay values

Change pattern	trace2vec		run [8]		process-graph [10]		Bose [3]	
	F-score	Mean delay	F-score	Mean delay	F-score	Mean delay	F-score	Mean delay
re	1	21.83	1	44.03	0.9036	33.02	1	17
cf	1	5.08	0.9824	21	0.9853	34.62	0.8950	36
lp	1	21.69	1	40.29	0.7618	48.03	0.6484	41
pl	1	13.8	1	35.74	0.9575	26.33	1	20
cb	1	9.1	0.9387	58.55	0.9722	18.94	0	0
cm	1	8.52	1	39.85	0.9722	19.24	0	0
cp	1	4.58	1	19.66	0.9853	17.59	0.6394	36
cd	1	46.44	0.8799	34.62	0.9546	28.62	1	20
pm	1	11.97	1	12.88	0.9869	24.78	0.7804	69
rp	1	4.86	0.9666	19.18	0.9722	12.67	0.75	40
sw	1	6.02	1	21.67	1	29.61	0.7804	39
fr	1	20.5	0.7569	49.92	0.9853	19.92	0.4420	165
IOR	1	36.66	1	19.11	0.9606	13.00	0.7804	38
IRO	1	6.8	1	43.96	0.9487	27.22	0.5611	82
OIR	0.9967	12.11	0.9803	47.89	0.7331	28.06	1	20
ORI	1	11.38	1	14.51	0.9869	14.25	0.7804	38
RIO	1	9.08	0.9824	23.81	0.9722	20.77	0.5611	60
ROI	1	7.66	1	22.51	1	7.31	1	20
Average	0.9998	12.15	0.9715	31.62	0.9466	23.56	0.7010	46.31

Table 4. The results of the t-tests. The first group is the trace2vec approach and the second group is either run, process-graph, or Bose approach

		Mean	Std. Deviation	Std. Error Mean	confidence		t	df	Sig. (2-tailed)
					Lower	Upper			
trace2vec-run	F-score	0.03000	0.06444	0.01519	-0.00205	0.06205	1.975	17	0.065
	Mean delay	-17.28333	16.66887	3.92889	-25.57257	-8.99410	-4.399	17	0.000
trace2vec-processGraph	F-score	0.05889	0.07324	0.01726	0.02247	0.09531	3.412	17	0.003
	Mean delay	-9.20833	13.66684	3.2213	-16.00469	-2.41197	-2.859	17	0.011
trace2vec-Bose	F-score	0.30000	0.30828	0.07266	0.14670	0.45330	4.129	17	0.001
	Mean delay	-31.28375	39.19712	9.799928	-52.17042	-10.39708	-3.192	15	0.006

5. Conclusion

This study proposed a new method for the detection of process drifts in business process logs. We introduced the new notions of trace embedding, which enabled us to surpass the state-of-the-art methods in the identification of predictable process drifts as well as unpredictable ones. Trace embedding can be used for automatic extracting of all features from the relations that exist between traces and for producing vector representations of traces. Thus, the relations that exist between traces in the log are represented by the relations that exist between vectors in the vector space. The experiments demonstrated that considering both F-score and mean delay, our approach is superior to the current methods. Moreover, these achievements have been made without using any types of windows.

In the future, we expect our study advance in the following ways:

1. This paper has dealt with process changes only from control-flow perspective. We plan to include changes from data and resource perspectives too;
2. The detection of sudden drift was addressed in this study. Likewise, detection of gradual and recurring drifts will be done;
3. This study plans to implement the approach as a ProM plug-in.

6. References

[1] J. C. Schlimmer and R. H. Granger, “Beyond incremental processing: Tracking concept drift.,” in AAAI, pp. 502–507, 1986.

[2] J. Martjushev, R. J. C. Bose, and W. M. Van Der Aalst, “Change point detection and dealing with gradual and multi-order dynamics in process mining,” in International Conference on Business Informatics Research, pp. 161–178, Springer, 2015.

[3] R. J. C. Bose, W. M. Van Der Aalst, I. Zliobaite, and M. Pechenizkiy, “Dealing with concept drifts in process mining,” IEEE transactions on neural networks and learning systems, vol. 25, no. 1, pp. 154–171, 2014.

[4] R. A. T. Stocker, “Discovering workflow changes with time-based trace clustering,”

[5] Lecture Notes in Business Information Processing, pp. 154–168, 2011.

[6] B. Hompes, J. C. Buijs, W. M. Van Der Aalst, P. Dixit, and H. Buurman, “Detect- ing change in processes using comparative trace clustering.,” in SIMPDA, pp. 95–108, 2015.

[7] J. Carmona and R. Gavalda, “Online techniques for dealing with concept drift in process mining,” in Proceedings of the 11th International Conference on Advances in Intelligent Data Analysis, IDA’12, pp. 90–102, Springer-Verlag, 2012.

[8] P. Weber, B. Bordbar, and P. Tino, “Real-time detection of process change using process mining.,” in ICCSW, pp. 108–114, 2011.

[9] A. Maaradji, M. Dumas, M. La Rosa, and A. Ostovar, “Fast and accurate business process drift detection,” in International Conference on Business Process Management, pp. 406–422, Springer, 2015.

[10] A. Ostovar, A. Maaradji, M. La Rosa, A. H. ter Hofstede, and B. F. van Don- gen, “Detecting drift from event streams of unpredictable business processes,” in Conceptual Modeling: ER 2016, pp. 330–346, Springer, 2016.

[11] A. Seeliger, T. Nolle, and M. Mu¨hlh¨auser, “Detecting concept drift in processes using graph metrics on process graphs,” in Proceedings of the 9th Conference on Subject-oriented Business Process Management, p. 6, ACM, 2017.

[12] A. Maaradji, M. Dumas, M. La Rosa, and A. Ostovar, “Detecting sudden and gradual drifts in business processes from execution traces,” IEEE Transactions on Knowledge and Data Engineering, vol. 29, no. 10, pp. 2140–2154, 2017.

[13] P. De Koninck, S. vanden Broucke, and J. De Weerdt, “act2vec, trace2vec, log2vec, and model2vec: Representation learning for business processes,” in Business Pro- cess Management, pp. 305–321, Springer International Publishing, 2018.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” arXiv preprint arXiv:1301.3781, 2013.

[15] M. Hassani, “Concept drift detection of event streams using an adaptive window.,” pp. 230–239, 2019.
[16] G. M. Tavares, P. Ceravolo, V. G. T. Da Costa, E. Damiani, and S. B. Junior, “Overlapping analytic stages in online process mining,” pp. 167–175, 2019.
[17] T. Li, T. He, Z. Wang, Y. Zhang, and D. Chu, “Unraveling process evolution by handling concept drifts in process mining,” in SCC, pp. 442–449, 2017.
[18] F. Stertz and S. Rinderle-Ma, “Process histories - detecting and representing con- cept drifts based on event streams,” in On the Move to Meaningful Internet Sys- tems. OTM 2018 Conferences (H. Panetto, C. Debruyne, H. A. Proper, C. A. Ardagna, D. Roman, and R. Meersman, eds.), pp. 318–335, Springer International Publishing, 2018.
[19] T. Mikolov, W.-t. Yih, and G. Zweig, “Linguistic regularities in continuous space word representations.,” in hlt-Naacl, vol. 13, pp. 746–751, 2013.
[20] R. J. C. Bose, W. M. Van der Aalst, I. Zˇliobaite˙, and M. Pechenizkiy, “Handling concept drift in process mining,” in International Conference on Advanced Infor- mation Systems Engineering, pp. 391–405, Springer, 2011.
[21] C. Zheng, L. Wen, and J. Wang, “Detecting process concept drifts from event logs,” pp. 524–542, 2017.
[22] Y. Spenrath and M. Hassani, “Ensemble-based prediction of business processes bottlenecks with recurrent concept drifts.,” 2019.
[23] Q. Le and T. Mikolov, “Distributed representations of sentences and documents,” in ICML, pp. 1188–1196, 2014.
[24] M. Rahman, Applications of Fourier transforms to generalized functions. WIT Press, 2011.
[25] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed repre- sentations of words and phrases and their compositionality,” in Advances in neural information processing systems, pp. 3111–3119, 2013.
[26] V. D. Aalst, Process Mining - Discovery, Conformance and Enhancement of Busi- ness Processes. Springer, 2011.
[27] W. Van Der Aalst, A. Adriansyah, A. K. A. De Medeiros, F. Arcieri, T. Baier, T. Blickle, J. C. Bose, P. van den Brand, R. Brandtjen, J. Buijs, et al., “Process mining manifesto,” in International Conference on Business Process Management, pp. 169–194, Springer, 2011.
[28] V. D. Aalst, M. L. Rosa, and F. M. Santoro, “Business process management - don’t forget to improve the process!,” Business & Information Systems Engineering, vol. 58, no. 1, pp. 1–6, 2016.
[29] M. Baroni, G. Dinu, and G. Kruszewski, “Don’t count, predict! a systematic com- parison of context-counting vs. context-predicting semantic vectors.,” in ACL (1), pp. 238–247, 2014.
[30] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, “Indexing by latent semantic analysis,” JOURNAL OF THE AMERICAN SOCI- ETY FOR INFORMATION SCIENCE, vol. 41, no. 6, pp. 391–407, 1990.
[31] A. Mandelbaum and A. Shalev, “Word embeddings and their use in sentence clas- sification tasks,” arXiv preprint arXiv:1610.08229, 2016.
[32] X. Rong, “word2vec parameter learning explained,” arXiv preprint arXiv:1411.2738, 2014.
[33] P. Ristoski and H. Paulheim, “Rdf2vec: Rdf graph embeddings for data mining,” in International Semantic Web Conference, pp. 498–514, Springer, 2016.
[34] R. J. C. Bose and W. M. Van der Aalst, “Context aware trace clustering: Towards improving process mining results,” in Proceedings of the 2009 SIAM International Conference on Data Mining, pp. 401–412, SIAM, 2009.
[35] R. J. C. Bose and W. M. Van der Aalst, “Trace clustering based on conserved pat- terns: Towards achieving better process models.,” in Business Process Management Workshops, vol. 43, pp. 170–181, Springer, 2009.
[36] G. Greco, A. Guzzo, L. Pontieri, and D. Sacca, “Discovering expressive process models by clustering log traces,” IEEE Transactions on Knowledge and Data En- gineering, vol. 18, no. 8, pp. 1010–1027, 2006.
[37] J. Demˇsar and Z. Bosni´c, “Detecting concept drift in data streams using model explanation,” Expert Systems with Applications, vol. 92, pp. 546–559, 2018.
[38] J. Evermann, J.-R. Rehse, and P. Fettke, “Predicting process behaviour using deep learning,” Decision Support Systems, 2017.
[39] T. S. Sethi and M. Kantardzic, “On the reliable detection of concept drift from streaming unlabeled data,” Expert Systems with Applications, vol. 82, pp. 77–99, 2017.
[40] T. Escovedo, A. Koshiyama, A. A. da Cruz, and M. Vellasco, “Detecta: abrupt concept drift detection in non-stationary environments,” Applied Soft Computing, vol. 62, pp. 119–133, 2018.
[41] A. Alves de Medeiros, B. Van Dongen, W. Van Der Aalst, and A. Weijters, “Process mining: Extending the alpha-algorithm to mine short loops,” tech. rep., BETA Working Paper Series, 2004.
[42] K. Fatemeh, “Concept drift detection in business process logs using deep learning,” Master’s thesis, Ferdowsi University of Mashhad, Iran, 2016.